

**Amendments to the Claims**

Please amend Claims 1, 4-6, 9, 11, 12, 13, 15-18, 22, 25, 28, 29, 32-35, 39, 42, 45, 46, 48-53. Please add new Claim 54. The Claim Listing below will replace all prior versions of the claims in the application:

**Claim Listing**

1. (Currently Amended) A computer implemented scheduling method comprising ~~the steps of:~~
  - based on scheduling states, defining a set of static schedules for an application, each static schedule including an assignment of tasks to processors;
  - during run time, learning a cost of a set of static schedules based on performance of the application; and
  - designating ~~the~~ a static schedule with ~~the~~ a lowest cost as an optimal schedule for ~~the~~ a scheduling state.
2. (Original) A scheduling method as claimed in Claim 1 wherein the cost of a set of static schedules is learned each time there is a change in scheduling state.
3. (Original) A scheduling method as claimed in Claim 1 wherein the cost of a set of static schedules is learned continuously during run time.
4. (Currently Amended) A scheduling method as claimed in Claim 1 further comprising ~~the steps of:~~
  - storing a set of all possible schedules associated with each schedule state; and
  - upon a change of state, selecting the optimal schedule associated with the schedule state.
5. (Currently Amended) A scheduling method as claimed in Claim 4 wherein ~~the~~ a selected schedule is the schedule with ~~the~~ a lowest cost.

6. (Currently Amended) A scheduling method as claimed in Claim 4 wherein ~~the~~ a selected schedule is the schedule with an unknown cost.
7. (Original) A scheduling method as claimed in Claim 6 wherein the schedule is randomly selected dependent on utility of exploration associated with the schedule.
8. (Original) A scheduling method as claimed in Claim 1 wherein the cost of a schedule is computed and stored after the schedule is executed.
9. (Currently Amended) A scheduling method as claimed in Claim 1 further comprising the ~~step of:~~  
maintaining a task execution cost for each task in the application for each scheduling state.
10. (Original) A scheduling method as claimed in Claim 9 wherein an optimal static schedule associated with a new scheduling state is computed using stored task execution costs.
11. (Currently Amended) A scheduling method as claimed in Claim 10 wherein the cost of an individual task is updated using ~~a sliding window which discounts older execution results at the expense of more recent execution results~~ stored task execution costs with recent schedule execution costs having more importance.
12. (Currently Amended) A scheduling method as claimed in Claim 10 wherein the cost of a schedule is updated using ~~a sliding window which discounts older execution results at the an expense of more recent execution results~~ stored task execution costs with recent schedule execution costs having more importance.
13. (Currently Amended) A scheduling method as claimed in Claim 1 further comprising the ~~step of:~~  
predicting the cost of a schedule dependent on stored task execution costs.

14. (Original) A scheduling method as claimed in Claim 13 wherein a schedule is selected for further exploration dependent on the predicted schedule cost.
15. (Currently Amended) A scheduling method as claimed in Claim 1 wherein the step of learning further comprises ~~the steps of~~:
  - storing application input data received during an active period in the application;
  - and
  - exploring optimal schedules while replaying ~~the~~ stored input data during an idle period in the application.
16. (Currently Amended) A scheduling method as claimed in Claim 15 wherein the step of learning further comprises ~~the step of~~:
  - concurrently executing a copy of an application with identical input data on a processor other than ~~the~~ another processor on which the application is executing.
17. (Currently Amended) A scheduling method as claimed in Claim 16 wherein a change in ~~the~~ optimized schedules is immediately reflected to ~~the~~ a schedule analyzer for use in ~~the~~ a next schedule change of the application.
18. (Currently Amended) A scheduling system comprising:
  - a set of static schedules for an application, the static schedules based on scheduling states, each static schedule including an assignment of tasks to processors;
  - and
  - a schedule analyzer which:
    - during run time, learns a cost of the set of static schedules based on performance of the application; and
    - designates ~~the~~ a static schedule with ~~the~~ a lowest cost as an optimal schedule for ~~the~~ a scheduling state.

19. (Original) A scheduling system as claimed in Claim 18 wherein the schedule analyzer learns the cost of a set of static schedules each time there is a change in scheduling state.
20. (Original) A scheduling system as claimed in Claim 18 wherein the schedule analyzer learns the cost of a set of static schedules continuously during run time.
21. (Original) A scheduling system as claimed in Claim 18 further comprising:  
a list of schedule costs which stores an optimal schedule associated with each schedule state wherein upon a change of state the schedule analyzer selects the optimal schedule corresponding to the schedule state.
22. (Currently Amended) A scheduling system as claimed in Claim 21 wherein the schedule analyzer selects a schedule with ~~the~~ a lowest cost.
23. (Original) A scheduling system as claimed in Claim 21 wherein the schedule analyzer selects a schedule with an unknown cost.
24. (Original) A scheduling system as claimed in Claim 23 wherein the schedule analyzer randomly selects a schedule dependent on utility of exploration associated with the schedule.
25. (Currently Amended) A scheduling system as claimed in Claim 18 wherein the schedule analyzer computes the cost of a schedule and stores ~~the~~ a computed cost after the schedule is executed.
26. (Original) A scheduling system as claimed in Claim 18 further comprises:  
a task execution table which stores a task execution cost for each task in the application for each scheduling state.

27. (Original) A scheduling system as claimed in Claim 26 wherein the schedule analyzer computes an optimal static schedule associated with a new scheduling state using stored task execution costs.
28. (Currently Amended) A scheduling system as claimed in Claim 27 wherein the schedule analyzer updates the cost of an individual task using a sliding window by discounting older execution results at ~~the~~ an expense of more recent execution results.
29. (Currently Amended) A scheduling system as claimed in Claim 27 wherein the schedule analyzer updates the cost of a schedule using a sliding window by discounting older execution results at ~~the~~ an expense of more recent execution results.
30. (Original) A scheduling system as claimed in Claim 18 wherein the schedule analyzer predicts the cost of a schedule dependent on stored task execution costs.
31. (Original) A scheduling system as claimed in Claim 30 wherein the scheduler analyzer selects a schedule for further exploration dependent on a predicted schedule cost.
32. (Currently Amended) A scheduling system as claimed in Claim 18 further comprising:  
memory which stores application input data received during an active period in the application, ~~the~~ stored application input data allowing the schedule analyzer to explore optimal schedules while replaying the application input data during an idle period in the application.
33. (Currently Amended) A scheduling system as claimed in Claim 32 wherein the schedule analyzer provides a copy of an application and the stored application input data for concurrent execution on a processor other than ~~the~~ another processor on which the application is executing.

34. (Currently Amended) A scheduling system as claimed in Claim 33 wherein a change in ~~the~~ optimized schedules is immediately reflected to the schedule analyzer for use in ~~the~~ a next schedule change of the application.
35. (Currently Amended) A scheduling system comprising:  
a set of static schedules for an application, the static schedules based on scheduling states, each static schedule including an assignment of tasks to processors;  
means for learning which during run time, learns a cost of a set of static schedules based on performance of the application; and  
means for selecting which designates ~~the~~ a static schedule with ~~the~~ a lowest cost as an optimal schedule for ~~the~~ a scheduling state.
36. (Original) A scheduling system as claimed in Claim 35 wherein the means for learning learns the cost of a set of static schedules is learned each time there is a change in scheduling state.
37. (Original) A scheduling system as claimed in Claim 35 wherein the means for learning learns the cost of a set of static schedules continuously during run time.
38. (Original) A scheduling system as claimed in Claim 35 further comprising:  
a list of schedule costs which stores an optimal schedule associated with each schedule state wherein upon a change of state the means for analyzing selects the optimal schedule associated with the schedule state.
39. (Currently Amended) A scheduling system as claimed in Claim 38 wherein the means for selecting selects a schedule with ~~the~~ a lowest cost.
40. (Original) A scheduling system as claimed in Claim 38 wherein the means for selecting selects a schedule with an unknown cost.

41. (Original) A scheduling system as claimed in Claim 40 wherein the means for selecting randomly selects a schedule dependent on utility of exploration associated with the schedule.
42. (Currently Amended) A scheduling system as claimed in Claim 35 wherein the means for selecting computes the cost of a schedule and stores ~~the~~ a computed cost after the schedule is executed.
43. (Original) A scheduling system as claimed in Claim 35 further comprises:  
a task execution table which stores a task execution cost for each task in the application for each scheduling state.
44. (Original) A scheduling system as claimed in Claim 43 wherein the means for selecting computes an optimal static schedule associated with a new scheduling state is using stored task execution costs.
45. (Currently Amended) A scheduling system as claimed in Claim 44 wherein the means for selecting updates the cost of an individual task using a sliding window by discounting older execution results at ~~the~~ a expense of more recent execution results.
46. (Currently Amended) A scheduling system as claimed in Claim 44 wherein the means for selecting updates the cost of a schedule using a sliding window by discounting older execution results at ~~the~~ a expense of more recent execution results.
47. (Original) A scheduling system as claimed in Claim 35 wherein the means for selecting predicts the cost of a schedule dependent on stored task execution costs.
48. (Currently Amended) A scheduling system as claimed in Claim 47 wherein the means for selecting selects a schedule for further exploration dependent on ~~the~~ a predicted cost for the schedule.

49. (Currently Amended) A scheduling system as claimed in Claim 35 wherein the ~~on-line scheduling system~~ further ~~comprises~~ comprising:  
memory which stores application input data received during an active period in the application, the ~~stored~~ application input data allowing the scheduling analyzer to explore optimal schedules while replaying the application input data during an idle period in the application.
50. (Currently Amended) A scheduling system as claimed in Claim 49 wherein ~~the~~ an on-line scheduling system provides a copy ~~of a copy~~ of an application and the ~~stored~~ application input data for concurrent execution on a processor other than ~~the~~ another processor on which the application is executing.
51. (Currently Amended) A scheduling system as claimed in Claim 18 wherein a change in ~~the~~ a optimized schedules is immediately reflected to the means for analyzing for use in ~~the~~ a next schedule change of the application.
52. (Currently Amended) A computer system comprising:  
a central processing unit connected to a memory system by a system bus;  
an I/O system, connected to the system bus by a bus interface; and  
a scheduling system routine located in the memory system which:  
based on scheduling states, defines a set of static schedules for an application, each static schedule including an assignment of tasks to processors;  
during run time, learns a cost of a set of static schedules based on performance of the application; and  
designates ~~the~~ a static schedule with ~~the~~ a lowest cost as an optimal schedule for ~~the~~ a scheduling state.



53. (Currently Amended) A computer program product for system scheduling, the computer program product comprising a computer usable medium having computer readable program code thereon, including program code which:

based on scheduling states, defines a set of static schedules for an application,  
each static schedule including an assignment of tasks to processors;

during run time, learns a cost of a set of static schedules based on performance of the application; and

designates ~~the~~ a static schedule with ~~the~~ a lowest cost as an optimal schedule for ~~the~~ a scheduling state.

54. (New) The method of claim 1, wherein the performance of the application is based on time to complete one iteration of the application.